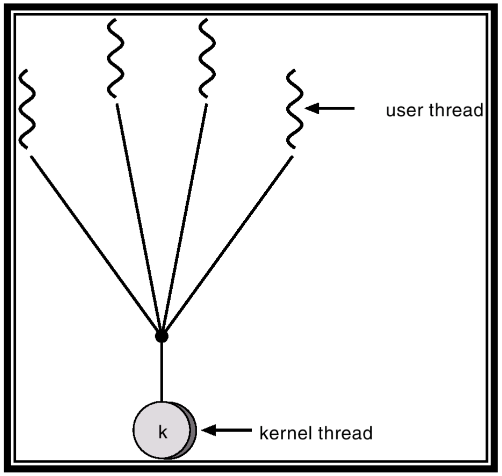
Thread types

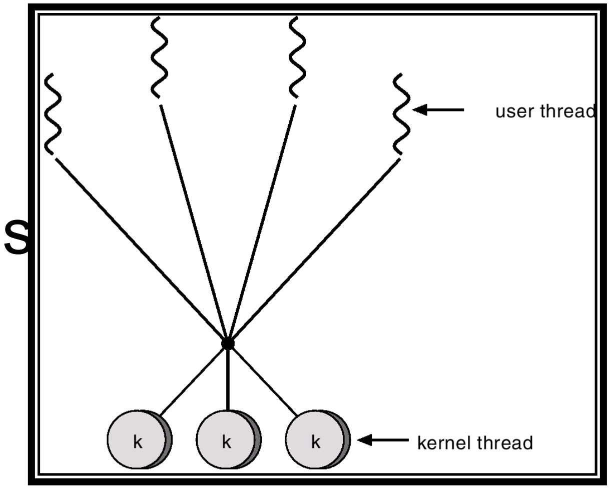
- Continuum: Cost to create and ease of management
- User level threads (e.g. pthreads)
  - Implemented as a library
  - Fast to create
  - Cannot have blocking system calls
  - Scheduling conflicts between kernel and threads. User level threads cannot do anything is kernel preempts the process
- Kernel level threads
  - Slower to create and manage
  - Blocking system calls are no problem
  - Most OS’s support these threads

Threading models

- One to One model
  - Map each user thread to one kernel thread
- Many to one model
  - Map many user threads to a single kernel thread
  - Cannot exploit multiprocessors



- Many to many
  - Map *m* user threads to *n* kernel threads



Threading Issues:

- Cancellation:
  - Asynchronous or deferred cancellation
- Signal handling:
  - Relevant thread
  - Every thread
  - Certain threads
  - Specific thread
- Pooled threads (web server)
- Thread specific data

Threads – Andrew Birrell

- Seminal paper on threads programming
  - Old but most techniques/experiences are still valid

- Birrell
  - Xerox PARC□ Dec SRC □ Microsoft Research
  - Invented Remote Procedure Calls (RPC)
  - Personal Juke box (hard disk based mp3 – Apple iPOD?)
  - Worked on Cedar, Distributed FS etc for ~25 years (1977-

Threads – Andrew Birrell

- Dec SRC and Xerox PARC were the premium systems research labs
- PARC researchers invented:
  - Personal computers - Alto
  - Mouse
  - Windows - Star
  - Bitmapped terminals
  - Icons
  - Ethernet
  - Smalltalk
  - Bravo – first WYSIWYG program
  - Laser printer
  - …

Windows 2000

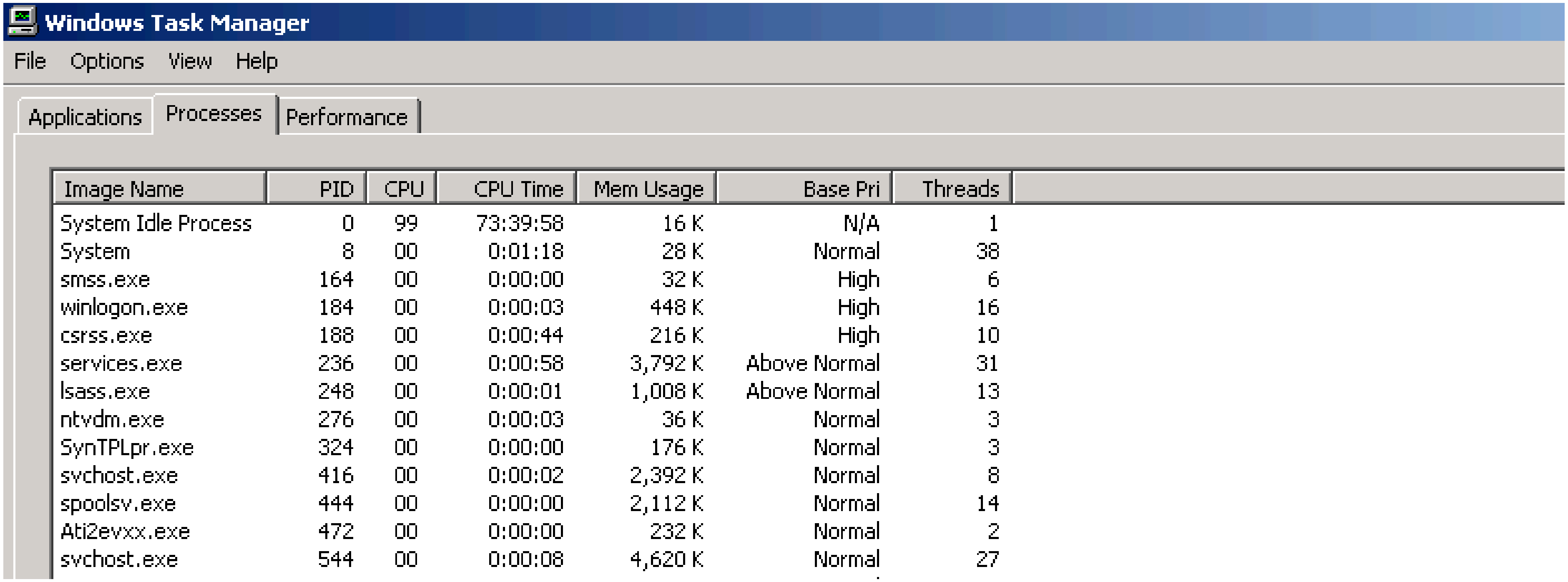


Image Name	PID	CPU	CPU Time	Mem Usage	Base Pri	Threads
System Idle Process	0	99	73:39:58	16 K	N/A	1
System	8	00	0:01:18	28 K	Normal	38
smss.exe	164	00	0:00:00	32 K	High	6
winlogon.exe	184	00	0:00:03	448 K	High	16
csrss.exe	188	00	0:00:44	216 K	High	10
services.exe	236	00	0:00:58	3,792 K	Above Normal	31
lsass.exe	248	00	0:00:01	1,008 K	Above Normal	13
ntvdm.exe	276	00	0:00:03	36 K	Normal	3
SynTPLpr.exe	324	00	0:00:00	176 K	Normal	3
svchost.exe	416	00	0:00:02	2,392 K	Normal	8
spoolsv.exe	444	00	0:00:00	2,112 K	Normal	14
Ati2evxx.exe	472	00	0:00:00	232 K	Normal	2
svchost.exe	544	00	0:00:08	4,620 K	Normal	27

Wizard 'ps -cfLeP' output

```
UID        PID   PPID   LWP  PSR    NLWP  CLS  PRI   STIME   TTY      LTIME  CMD
 root       0      0      1    -      1   SYS   96   Aug 03 ?          0:01  sched
 root       1      0      1    -      1    TS   59   Aug 03 ?          7:12  /etc/init -
 root       2      0      1    -      1   SYS   98   Aug 03 ?          0:00  pageout
 root       3      0      1    -      1   SYS   60   Aug 03 ?        275:46  fsflush

root     477    352      1    -      1    IA   59   Aug 03 ??        0:0
                                                       /usr/openwin/bin/fbconsole -d :0
 root      62      1     14    -     14    TS   59   Aug 04 ?          0:00
                                                            /usr/lib/sysevent/syseventd
```

Discussion

- Constant tension between moving functionality to upper layers; involving the application programmer and performing automatically at the lower layers
- Automatically create/manage threads by compiler/ system? (open research question)

CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state.
  2. Switches from running to ready state.
  3. Switches from waiting to ready.
  4. Terminates.
- Scheduling under 1 and 4 is *nonpreemptive*.
- All other scheduling is *preemptive.*

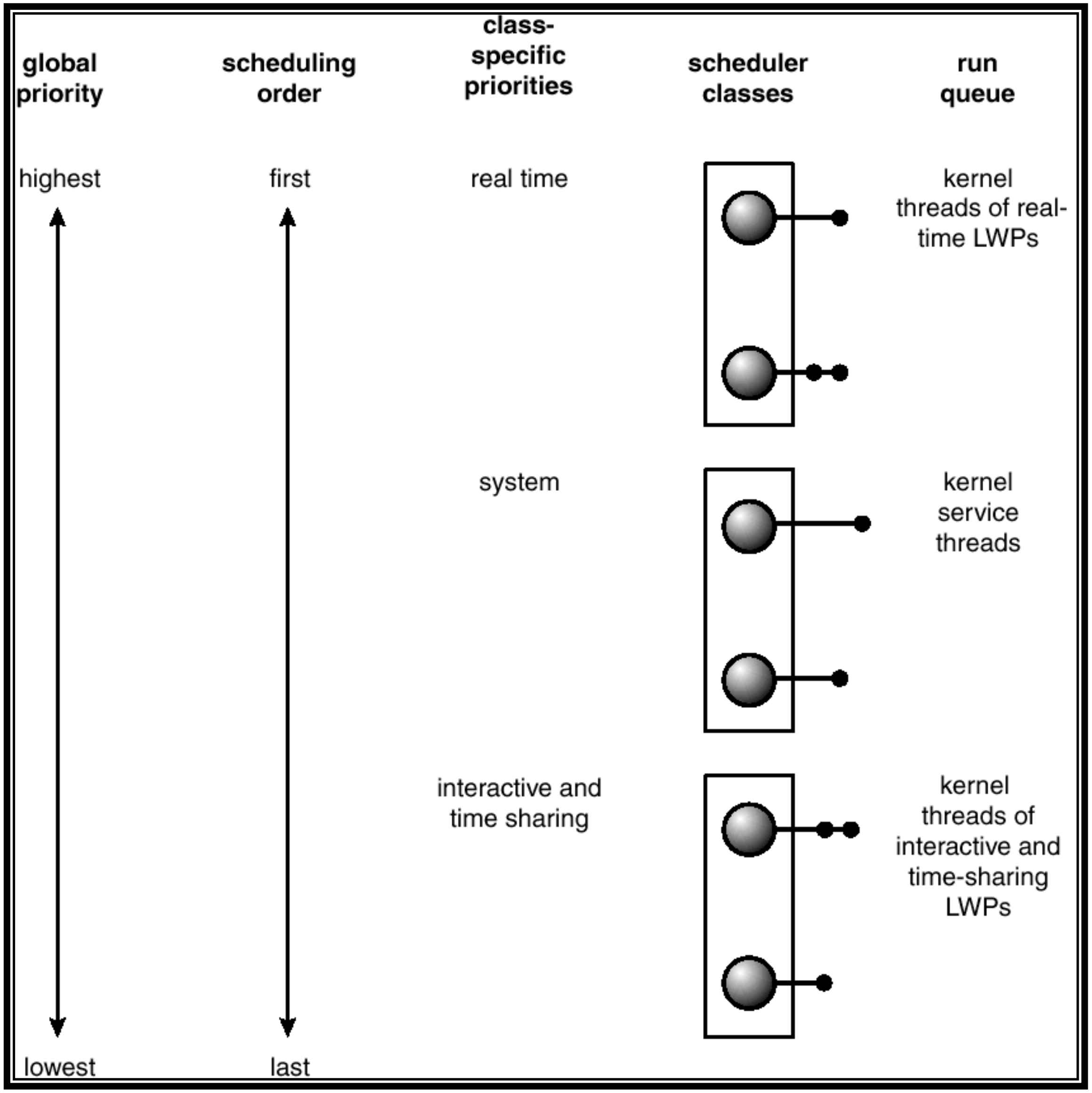
Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process
- Waiting time – amount of time a process has been waiting in the ready queue
- Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

CPU Scheduling - Classical algorithms

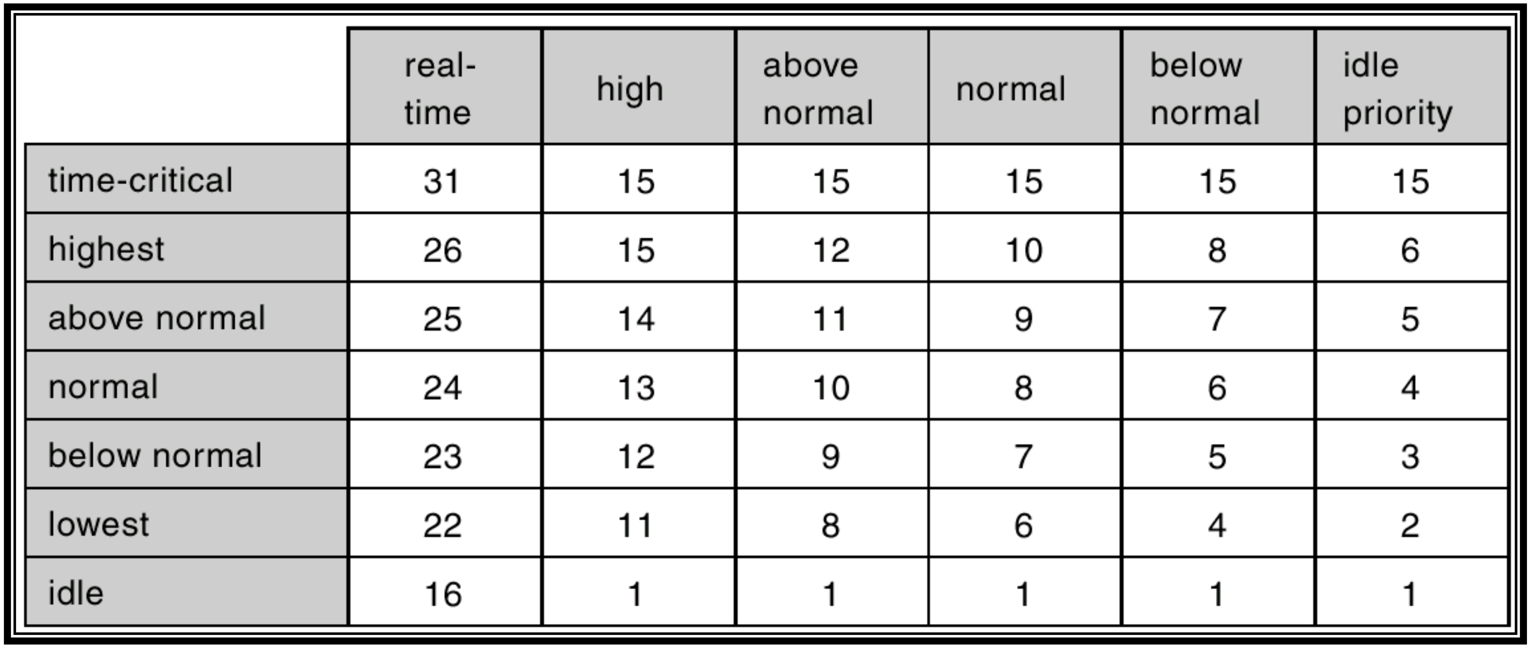
- First come, first served (FCFS)
  - Convoy effect (all processes get bunched up behind long process)
- Shortest job first (shortest next CPU burst)
  - “Optimal”.
  - Need oracle to predict next duration: prediction based on history
- Priority: Choose higher priority tasks
  - Priority inversion: High priority tasks wait for lower priority tasks holding critical resources
  - Starvation: Low priority tasks never get their chance
- Round robin
- Multilevel queue: Different schemes for background, interactive tasks
- Multilevel feedback queue scheduling
- Multiprocessor scheduling
  - Gang scheduling, Non Uniform Memory Access, Processor affinity
- Real-Time scheduling
  - Hard and soft real time systems

Solaris 2 Scheduling



global priority
scheduling order
class-specific priorities
scheduler classes
run queue
highest
first
real time
kernel threads of real-time LWPs
system
kernel service threads
interactive and time sharing
kernel threads of interactive and time-sharing LWPs
lowest
last

Windows 2000 Priorities



	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Tools to play with

- FreeBSD: rtprio, idprio
- Windows XP: Task manager
- Solaris: prioctl
- UNIX: nice, renice, setpriority, ps, top
- Procfs - a pseudo file system for process stats
  - Look at files in /proc/curproc in FreeBSD, /proc in Linux, /proc in Solaris